PERANCANGAN FUZZY FILTER DENGAN FPGA

Satrio Dewanto, Susido, Chandra Gunawan, Julianto

Jurusan Sistem Komputer, Universitas Bina Nusantara
Email :
satrio@binus.ac.id, susido_cheung@yahoo.com,
chandragunawan2000@yahoo.com, nick_bestoo@yahoo.com.

Abstrak

Pada proses pengambilan citra, sering terjadi citra yang diperoleh mengandung noise yang mengganggu tampilan visual citra. Agar tampilan citra menjadi jelas, maka noise yang menyertai citra perlu dihilangkan. Pada makalah ini akan digunakan logika fuzzy untuk menghilangkan noise yang ada pada suatu citra. Citra yang digunakan dibatasi pada citra keabuan (grayscale image) yang telah ditambahi dengan noise. Proses penapisan noise menggunakan logika fuzzy yang terdiri dari perhitungan fuzzy derivative dan fuzzy smoothing. Penapisan dengan logika fuzzy ini akan dilaksanakan secara perangkat keras menggunakan field programmable gate array (FPGA). Hasil penelitian memperlihatkan bahwa proses penapisan secara perangkat keras ini dapat berjalan dengan baik dimana noise yang menyertai citra dapat dihilangkan sehingga tampilan visual menjadi lebih jelas.Dari hasil percobaan dapat disimpulkan bahwa waktu proses dengan FPGA lebih cepat dibandingkan dengan menggunakan perangkat lunak. Hasil penelitian ini akan memberikan peluang untuk pengembangan embedded system dimana fuzzy filter dengan FPGA digunakan sebagai bagian dari suatu sistem pengolahan citra.

Kata Kunci : fuzzy filter, FPGA, derau, logika fuzzy

1. Pendahuluan

Aplikasi dari teknik-teknik *fuzzy* dalam pengolahan citra merupakan bidang penelitian yang menjanjikan [1]. Teknik-teknik *fuzzy* telah diterapkan dalam beberapa domain dari pengolahan citra (seperti pem-*filter*-an, interpolasi, dan morphologi) dan beberapa aplikasi yang praktis (seperti pengolahan citra dalam industri dan medis).

Rancangan *fuzzy filter* dalam thesis ini difokuskan pada teknik *fuzzy* untuk pem-*filter*-an citra. Teknik-teknik *fuzzy* untuk pem-*filter*-an citra yang banyak dikembangkan adalah *fuzzy filter* untuk *noise reduction*. Beberapa *fuzzy filter* yang telah dikembangkan untuk *noise reduction* seperti FIRE-*filter*, *Weighted Fuzzy Mean Filter* (WFMF), dan *filter* berdasarkan kontrol *fuzzy* yang iteratif. Kebanyakan teknik-teknik *fuzzy* yang dikembangkan tidak dirancang secara khusus untuk *gaussian noise* atau tidak menghasilkan nilai yang meyakinkan ketika diaplikasikan untuk melakukan pem-*filter*-an terhadap *gaussian noise*.

Dua tahapan penting dari *fuzzy filter* adalah: pertama, *filter* memperkirakan *fuzzy* derivatif untuk mengurangi sensitivitas dari variasi-variasi lokal yang berhubungan dengan struktur citra seperti *edge*; kedua, fungsi-fungsi keanggotaan diadaptasikan menurut tingkat *noise* untuk melakukan *fuzzy smoothing*.

Untuk masing-masing pixel yang telah diproses, tahapan pertama menghitung *fuzzy* derivatif. Kedua, kumpulan dari 16 aturan *fuzzy* diterapkan untuk mencari *correction term*. Aturan-aturan tersebut menggunakan *fuzzy* derivatif sebagai input. Himpunan *fuzzy* diterapkan untuk merepresentasikan karakteristik `small`, `positive` dan `negative`.

2. Fuzzy Filter

Gambaran umum dari *filter* adalah merata-ratakan *pixel* menggunakan nilai-nilai *pixel* yang bersebelahan, dengan memperhatikan struktur yang penting dari citra seperti *edge*. Perhatian utama

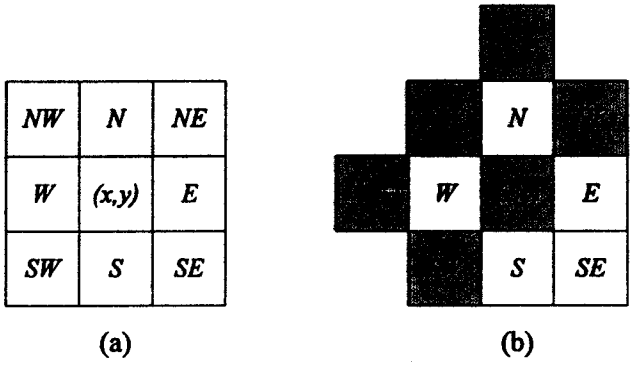
dari *filter* ini adalah membedakan variasi-variasi lokal yang berhubungan dengan *noise* dan struktur citra.

Untuk masing-masing *pixel* dihitung selisih dari nilai *pixel* yang menyatakan tingkatan dimana derivatif dalam arah tertentu adalah kecil. Misalkan nilai yang telah dikurangi untuk masing-masing arah berhubungan dengan *pixel-pixel* yang bersebelahan dari *pixel* yang diproses oleh aturan *fuzzy* (Bagian II.A).

Rancangan *filter* lebih lanjut didasarkan pada observasi bahwa *fuzzy* derivatif yang kecil lebih disebabkan oleh *noise*, sedangkan *fuzzy* derivatif yang besar lebih disebabkan oleh *edge* pada citra. Untuk masing-masing arah diterapkan dua aturan *fuzzy* untuk membedakan variasi-variasi lokal yang berhubungan dengan *noise* dan struktur citra, dan menghitung kontribusi dari nilai-nilai *pixel* yang bersebelahan. Hasil dari 16 aturan *fuzzy* tersebut di-defuzzifikasi dan *correction term* diperoleh untuk nilai *pixel* yang telah diproses (Bagian II.B).

A. *Fuzzy* Derivatif

Pendekatan pertama dari *fuzzy filter* adalah melakukan perhitungan *edge*. Untuk mencari *edges*, dapat dibuat perkiraan yang baik dengan menetapkan aturan-aturan *fuzzy*.



Gambar 1. (a) Pusat *pixel* (x, y) dengan *pixel* yang bersebelahan. (b) Nilai *pixel* yang ditandai dengan warna hitam digunakan untuk menghitung *fuzzy* derivatif dari pusat *pixel* (x, y) untuk arah *NW*

Tabel 1. *Pixel-pixel* yang digunakan untuk menghitung *fuzzy* derivatif pada masing-masing arah

Arah	Posisi	Himpunan w, r, t (x, y)
NW	$(x-1, y-1)$	{(-1, 1), (0, 0), (1, -1)}
W	$(x-1, y)$	{(0, 1), (0, 0), (0, -1)}
SW	$(x-1, y+1)$	{(1, 1), (0, 0), (-1, -1)}
S	$(x, y+1)$	{(1, 0), (0, 0), (-1, 0)}
SE	$(x+1, y+1)$	{(1, -1), (0, 0), (-1, 1)}
E	$(x+1, y)$	{(0, -1), (0, 0), (0, 1)}
NE	$(x+1, y-1)$	{(-1, -1), (0, 0), (1, 1)}
N	$(x, y-1)$	{(-1, 0), (0, 0), (1, 0)}

Ambil *pixel* (x, y) yang bersebelahan 3x3, seperti yang ditampilkan pada Gambar 1(a).

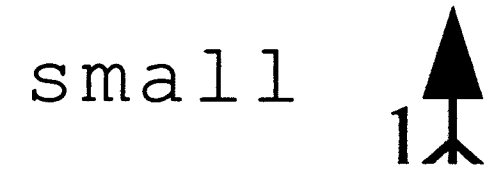
Derivatif yang sederhana pada posisi pusat *pixel* (x, y) dalam arah D ($D \in dir = \{NW, W, SW, S, SE, E, NE, N\}$) didefinisikan sebagai selisih antara *pixel* pada titik (x,y) dan *pixel-pixel* disebelahnya dalam arah D. Nilai derivatif dinotasikan dengan $\nabla_D(x,y)$. Sebagai contoh :

$$\nabla_N(x,y) = I(x,y-1) - I(x,y)$$

$$\nabla_{NW}(x,y) = I(x-1,y-1) - I(x,y). \quad (1)$$

Prinsip-prinsip dari *fuzzy* derivatif adalah sebagai berikut: Misalkan *edge* yang melewati pusat *pixel* (x, y) dengan *pixel* bersebelahan dalam arah *SW – NE*. Nilai derivatif $\nabla_{NW}(x,y)$ akan besar, tetapi nilai-nilai derivatif dari *pixel-pixel* yang bersebelahan juga dapat diperkirakan besar. Sebagai contoh, dalam arah NW, dapat dihitung nilai-nilai $\nabla_{NW}(x,y)$, $\nabla_{NW}(x-1,y+1)$ dan $\nabla_{NW}(x+1,y-1)$ [Gambar 1(b)]. Ide ini untuk menghilangkan efek dari salah satu nilai derivatif yang menghasilkan nilai yang tinggi yang berhubungan dengan *noise*. Untuk itu, jika dua dari tiga derivatif bernilai kecil, maka dapat diasumsikan tidak ada *edge* dalam arah yang ditentukan. Berdasarkan perhitungan di atas, dapat dirumuskan aturan *fuzzy* untuk menghitung nilai-nilai *fuzzy* derivatif.

Tabel 1 berisi ringkasan dari *pixel* yang digunakan untuk menghitung *fuzzy* derivatif untuk masing-masing arah. Kolom-1 mewakili arah. Kolom-2 mewakili posisi dari arah yang ditetapkan pada kolom-1 terhadap titik pusat (x, y). Kolom-3 digunakan sebagai posisi *pixel* yang mewakili nilai pusat (x, y) dari posisi tersebut.



Gambar 2. Fungsi keanggotaan (a) `small`, (b) `positive`, dan (c) `negative`

Derajat keanggotaan dari *fuzzy derivatif* dalam arah tertentu adalah kecil, maka digunakan himpunan *fuzzy* `small`. Fungsi keanggotaan $m_K(u)$ untuk `small` adalah sebagai berikut [Gambar 2(a)]:

$$m_K(u) = \begin{cases} 1 - \dfrac{|u|}{K} & 0 \le |u| \le K \\ 0 & |u| > K \end{cases}. \quad (2)$$

dimana K adalah parameter adaptif.

Sebagai contoh, nilai-nilai dari *fuzzy* derivatif $\nabla^F_{NW}(x,y)$ untuk *pixel* (x, y) dalam arah *NW* dapat dicari dengan menetapkan aturan-aturan sebagai berikut :

JIKA ($\nabla_{NW}(x,y)$ adalah `small` DAN $\nabla_{NW}(x-1,y+1)$ adalah `small`) ATAU
($\nabla_{NW}(x,y)$ adalah `small` DAN $\nabla_{NW}(x+1,y-1)$ adalah `small`) ATAU
($\nabla_{NW}(x-1,y+1)$ adalah `small` DAN $\nabla_{NW}(x+1,y-1)$ adalah `small`)
MAKA $\nabla^F_{NW}(x,y)$ adalah `small`. (3)

Delapan aturan yang sama seperti diatas diterapkan untuk menghitung derajat keanggotaan dari *fuzzy* derivatif $\nabla^F_D(x,y)$, $D \in dir$, untuk himpunan `small`. Aturan- aturan tersebut diimplementasikan dengan minimum untuk mewakili operator-DAN, dan maksimum untuk mewakili operator-ATAU.

B. *Fuzzy Smoothing*

Untuk menghitung nilai *correction term* Δ untuk nilai *pixel* yang diproses, digunakan sepasang aturan *fuzzy* untuk masing-masing arah. Dari hasil perhitungan *fuzzy* derivatif, perlu dibedakan antara nilai-nilai positif dan negatif.

Sebagai contoh, untuk arah *NW*. Dengan nilai-nilai $\nabla^F_{NW}(x,y)$ dan $\nabla_{NW}(x,y)$, diterapkan 2 aturan untuk menghitung nilai kebenaran λ^+_{NW} dan λ^-_{NW} :

λ^+_{NW} : JIKA $\nabla^F_{NW}(x,y)$ adalah `small` DAN $\nabla_{NW}(x,y)$ adalah `positive`
MAKA c adalah `positive`
λ^-_{NW} : JIKA $\nabla^F_{NW}(x,y)$ adalah `small` DAN $\nabla_{NW}(x,y)$ adalah `negative`
MAKA c adalah `negative`. (4)

Enam belas aturan yang sama seperti diatas diterapkan untuk menghitung setiap *fuzzy smoothing* λ^+_D dan λ^-_D, $D \in dir$. Untuk fungsi keanggotaan `positive` dan `negative`, digunakan fungsi keanggotaan yang linear [Gambar 2(b) dan (c)]. Aturan- aturan tersebut diimplementasikan dengan minimum untuk mewakili operator-DAN, dan maksimum untuk mewakili operator-ATAU.

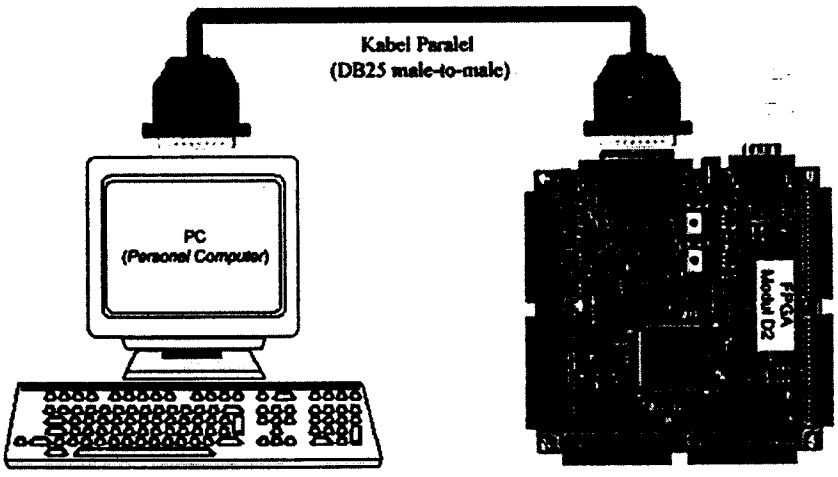
Tahapan terakhir dari perhitungan *fuzzy filter* adalah defuzzifikasi. Defuzzifikasi dalam *fuzzy filter* yaitu mencari nilai *correction term* Δ, yang akan ditambahkan ke nilai dari *pixel* pada titik (x, y). Nilai kebenaran dari aturan-aturan λ^+_D dan λ^-_D, $D \in dir$ (untuk semua arah) dikumpulkan dengan menghitung rata-rata kebenaran sebagai berikut :

$$\Delta = \frac{L}{8} \sum_{D \in dir} (\lambda^+_D - \lambda^-_D). \quad (5)$$

dimana *dir* adalah arah dan L adalah bilangan dari level *grayscale*. Jadi, setiap arah berkontribusi terhadap *correction term* Δ.

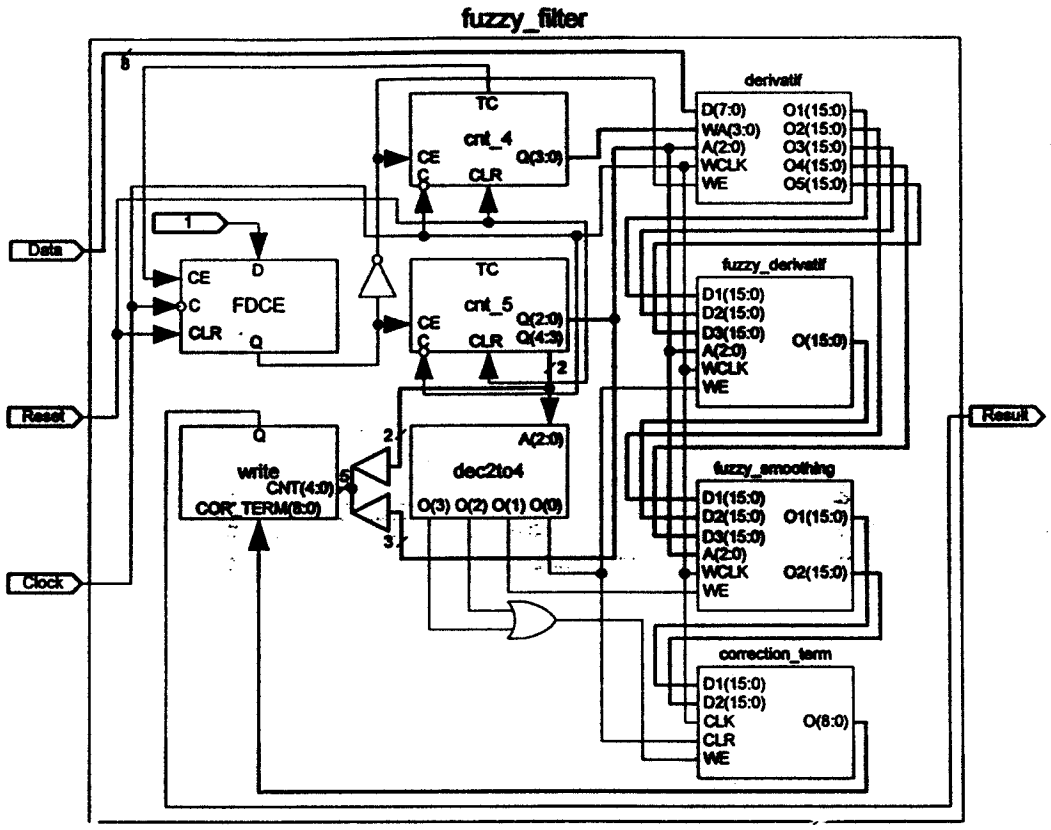
3. Perancangan Sistem

Perancangan *fuzzy filter* dengan FPGA ini terdiri dari sebuah PC *(Personal Computer)* dan modul FPGA Xilinx Spartan 2 XC2S200-PQ208 [Gambar (3)]. Antar muka antara PC dengan FPGA menggunakan kabel paralel (DB25 *male to male*). *Input/output* dari sistem dilakukan oleh PC sedangkan proses dari sistem dilakukan oleh FPGA.



Gambar 3. Rancangan Sistem

Rancangan skematik dari *fuzzy filter* dengan FPGA mempunyai 3 buah *input* dan sebuah *output* [Gambar 4.].



Gambar 4. Rancangan skematik dari *fuzzy filter*.

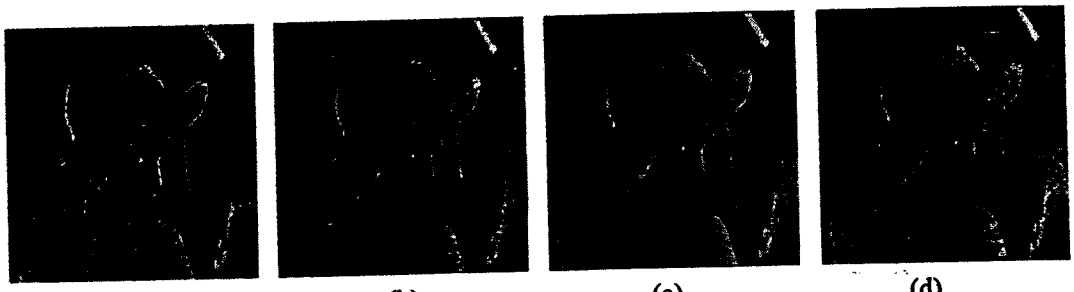
Input dari *fuzzy filter* adalah 8-*bit* `Data`, 1-*bit* `Reset`, dan 1-*bit* `Clock`. Sedangkan *output* dari *fuzzy fitler* adalah 1-*bit* `Result`. Untuk melakukan *reset* pada *fuzzy filter*, dengan mengirim sinyal HIGH ke pin `reset`. Setelah melakukan *reset*, maka *data* siap dikirim disertai dengan mengirim sinyal *clock* eksternal. *Counter*-4 akan bekerja, sehingga *data* yang diterima ditempatkan pada *address* yang tepat pada RAM. Setelah semua *data* telah selesai dikirim maka, maka sinyal `TC` pada *counter*-4 akan mengaktifkan *counter*-5 dan menon-aktifkan *counter*-4 itu sendiri. 3-*bit* dari *counter*-5 merupakan *input address* pada *memory* dan 2-*bit* lainnya merupakan *control unit* dari sistem. *Control unit* berfungsi untuk mengaktifkan komponen agar dapat bekerja. Setelah semua data telah selesai diproses, maka *output* yang dihasilkan akan dikirim secara *serial* melalui *pin* `Result`.

4. Hasil Percobaan

Fuzzy filter yang dirancang ini diterapkan pada citra *grayscale* (8-bit, $L = 255$), setelah ditambahkan *Gaussian noise* pada tingkat yang berbeda. Beberapa prosedur dibandingkan dan dievaluasi antara citra yang asli dengan citra yang telah di-*filter*. Gambar 5 adalah citra asli "Lena".

 ISSN : 1411-6286

Gambar 5. Citra asli "Lena"

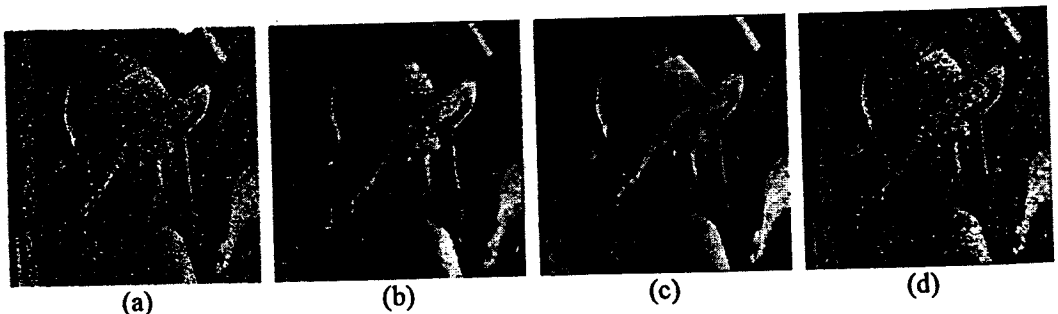


(a) (b) (c) (d)

Gambar 6. (a) "Lena" dengan *gaussian noise* 5% ($\sigma = 5$). (b) *fuzzy filter* FPGA. (c) *median filter*. (d) *mean filter*

(a) (b) (c) (d)

Gambar 7. (a) "Lena" dengan *gaussian noise* 10% ($\sigma = 10$). (b) *fuzzy filter* FPGA. (c) *median filter*. (d) *mean filter*



(a) (b) (c) (d)

Gambar 8. (a) "Lena" dengan *gaussian noise* 20% ($\sigma = 20$). (b) *fuzzy filter* FPGA. (c) *median filter*. (d) *mean filter*

Fuzzy filter yang dirancang dibandingkan dengan beberapa teknik *filter* yang lain seperti *median filter* dan *mean filter*. Pengolahan citra dengan noise dilakukan menggunakan perangkat lunak visual basic untuk *median filter, mean filter* dan *fuzzy filter*. Proses fuzzy filter ini juga dilakukan secara perangkat keras menggunakan FPGA selain menggunakan perangkat lunak. Hasil fuzzy filter

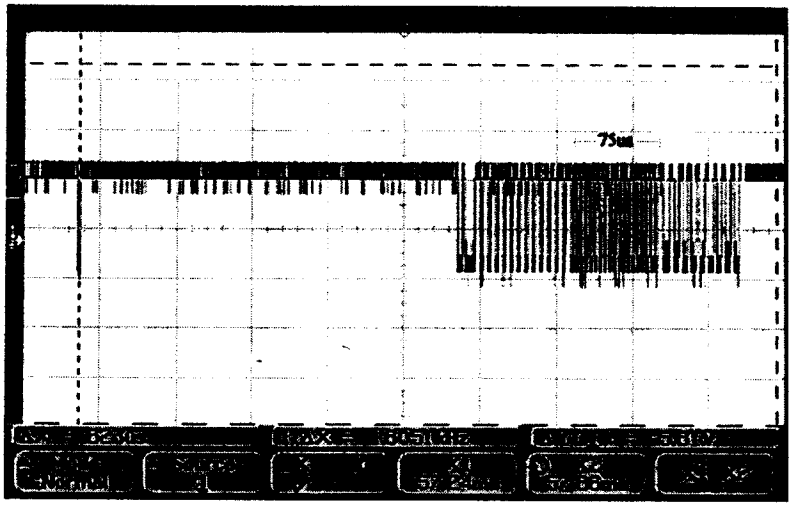
dengan perangkat lunak tidak ditampilkan pada gambar diatas karena tidak terlalu berbeda dengan hasil fuzzy filter menggunakan FPGA.

Tabel 2 menunjukkan perbandingan MSE antara *fuzzy filter* yang telah dirancang dengan *filter* yang lain seperti *mean filter* dan *median filter*. Nilai MSE terkecil didapat oleh pada *median filter*.

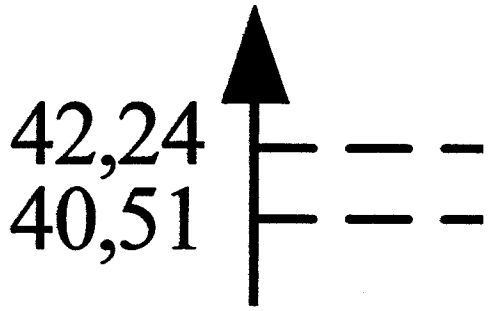
Tabel 2. Hasil yang diperoleh dari *fuzzy filter* untuk citra "Lena"

Jenis *Filter*	MSE		
	$\sigma = 5$	$\sigma = 10$	$\sigma = 20$
Fuzzy Filter FPGA (*hardware*)	107,54	155,68	181,74
Fuzzy Filter (software)	129,27	139,65	180,31
Median Filter (software)	76,97	84,97	106,42
Mean Filter (software)	152,94	207,44	308,56

Untuk melakukan proses *fuzzy filter* terhadap 1-*pixel* pada FPGA dibutuhkan waktu 623 us (*micro second*) [Gambar 9(a)], sehingga untuk melakukan proses terhadap 1-citra dengan ukuran 255x255 *pixel* dibutuhkan waktu 40,51 s (*second*) [Gambar 9(b)]. Untuk melakukan proses terhadap 1-*pixel*, FPGA melakukan komunikasi data dengan komputer dalam melakukan pengiriman data yang akan diproses pada FPGA. Spesifikasi komputer yang digunakan adalah prosesor dengan frekuensi 450 MHz (*Mega Hertz*), *memory* 392 MHz, dan sistem operasi Windows XP. Jika *fuzzy filter* FPGA diterapkan sebagai *embedded system* maka untuk melakukan proses terhadap 1-*pixel* dibutuhkan waktu 75 us [Gambar 9(a)], sehingga untuk melakukan proses terhadap 1-citra dengan ukuran 255x255 *pixel* hanya membutuhkan waktu 4,88 s [Gambar 9(b)] dengan frekuensi eksternal *clock* yang digunakan adalah 240 KHz (*Kilo Hertz*).



(a)



(b)

Gambar 9. Waktu eksekusi (a) *Fuzzy Filter* FPGA per *pixel*. (b) *Fuzzy Filter*

Untuk melakukan proses *fuzzy filter* terhadap 1-*pixel* pada FPGA dibutuhkan waktu 623 us (*micro second*) [Gambar 9(a)], sehingga untuk melakukan proses terhadap 1-citra dengan ukuran 255x255 *pixel* dibutuhkan waktu 40,51 s (*second*) [Gambar 9(b)]. Untuk melakukan proses terhadap 1-*pixel*, FPGA melakukan komunikasi data dengan komputer dalam melakukan pengiriman data yang akan diproses pada FPGA. Spesifikasi komputer yang digunakan adalah prosesor dengan frekuensi 450 MHz (*Mega Hertz*), *memory* 392 MHz, dan sistem operasi Windows XP. Jika *fuzzy filter* FPGA diterapkan sebagai *embedded system* maka untuk melakukan proses terhadap 1-*pixel* dibutuhkan waktu 75 us [Gambar 9(a)], sehingga untuk melakukan proses terhadap 1-citra dengan ukuran 255x255 *pixel* hanya membutuhkan waktu 4,88 s [Gambar 9(b)] dengan frekuensi eksternal *clock* yang digunakan adalah 240 KHz (*Kilo Hertz*).

5. Kesimpulan

Makalah ini mengusulkan perancangan *fuzzy filter* menggunakan FPGA. Hasil percobaan menunjukkan kelayakan dari *fuzzy filter* yang telah dirancang bila dibandingkan dengan teknik filter yang lain. Ukuran numerik, seperti MSE dan pengamatan *visual* menunjukkan bahwa *fuzzy filter* memberikan hasil yang cukup memuaskan walaupun MSE median filter lebih kecil tetapi dari penampakan visual, citra dengan median filter agak sedikit *blur*.

Waktu eksekusi *fuzzy filter* dengan FPGA menunjukkan hasil yang menjanjikan dibandingkan waktu eksekusi menggunakan perangkat lunak. Rancangan *fuzzy filter* yang cukup sederhana ini sangat mudah diimplementasikan pada perangkat keras, seperti FPGA. Hasil penelitian ini akan memberikan peluang untuk pengembangan embedded system dimana fuzzy filter dengan FPGA digunakan sebagai bagian dari suatu sistem pengolahan citra.

6. Daftar Pustaka

[1] E. Kerre, dan M. Nachtegael, *Fuzzy Techniques in Image Processing*. New York: Springer-Verlag, 2000, vol. 52, Studies in Fuzziness and Soft Computing.

[2] D. Van De Ville, M. Nachtegael, D. Van der Weken, E. Kerre, W. Philips, dan I. Lemahieu, "Noise Reduction by Fuzzy Image Filtering," *IEEE Transactions on Fuzzy Systems*, vol. 11, pp. 429-435, Aug. 2003.

[3] Mano, M. Morris and Kime, Charles R., *Logic and Computer Design Fundamentals*, Prentice-Hall International, Inc.